

# Algoritmos e Técnicas de Programação

Edwaldo Soares Rodrigues

Departamento de Ciência da Computação / Sistemas de Informação

PUC Minas São Gabriel

2023/1

# Classes

- **Classe:** é uma estrutura para representar uma Entidade;
- Uma classe pode conter:
  - *Atributos (Variáveis);*
  - *Métodos (Ações);*
- Por exemplo: **classe Pessoa**
  - *Possui os atributos: Nome, Endereço e Telefone;*
  - *Pode executar os métodos: Andar e Falar;*
- Uma classe pode ser vista como um tipo composto;



# Representação de uma Classe

```
class <nome_da_classe>
{
    //declaração dos atributos
    <visibilidade private/public> <tipo> <nome da variável>;

    //declaração de métodos
}
```



# Representação de uma Classe

```
class Pessoa
{
    //declaração dos atributos
    private string nome;
    public string endereco;

    public void InformarDados(string novoNome, string novoEndereco )
    {
        nome = novoNome;
        endereco = novoEndereco;
    }
    public string ObterNome()
    {
        return nome;
    }
}
```



# Objeto

- É um elemento de uma classe;
- Por exemplo: **fulano\_de\_tal** é um elemento da classe **Pessoa**;
- Em outras palavras: um objeto é uma variável do tipo da Classe (ou instância da classe);
- Como declarar um objeto de uma classe?
  - <Nome da Classe> <nome do objeto> = new <Construtor(>);
  - Pessoa fulano\_de\_tal = new Pessoa();



# Acessando os métodos e atributos do objeto

- Os métodos e atributos do objeto são acessados com ponto (.):
  - nomeObjeto.nomeMetodo();
  - nomeObjeto.nomeAtributo;

```
class Program{  
    public static void Main(){  
        Pessoa fulano_de_tal = new Pessoa();  
  
        fulano_de_tal.InformarDados("Fulano de Tal", "Rua setenta, 70 - Bairro - CEP");  
  
        Console.WriteLine(fulano_de_tal.endereco);  
  
        Console.WriteLine(fulano_de_tal.ObterNome());  
        //fulano_de_tal.nome; -> Erro:  
        //nome não é acessível devido ao nível de proteção  
    }  
}
```



# Acessando os métodos e atributos do objeto

- Somente métodos e atributos públicos (public) podem ser acessados fora da classe;

```
class Pessoa{  
    //declaração de variável  
    2 references  
    private string nome;  
    2 references  
    public string endereco;  
    1 reference  
    public void InformarDados(string novoNome, string novoEndereco ){  
        nome = novoNome;  
        endereco = novoEndereco;  
    }  
    1 reference  
    public string ObterNome(){  
        return nome;  
    }  
}  
0 references  
class Program{  
    0 references  
    public static void Main(){  
        Pessoa fulano_de_tal = new Pessoa();  
        fulano_de_tal.InformarDados("Fulano de Tal", "Rua X");  
        Console.WriteLine(fulano_de_tal.endereco);  
        Console.WriteLine(fulano_de_tal.ObterNome());  
        //fulano_de_tal.nome; -> Erro:  
        //nome não é acessível devido ao nível de proteção  
    }  
}
```



# Exercício

- Crie uma classe Agenda que armazena 5 pessoas (em um vetor), a classe deve ter:
  - Um método para inserir, uma nova pessoa, esse método deve receber o nome e o endereço da pessoa;
  - Um método que retorne a pessoa de uma cada posição do vetor de pessoas;





# Exercício

```
class Pessoa{  
    private string nome;  
    public string endereco;  
    public void InformarDados(string novoNome, string novoEndereco ){  
        nome = novoNome;  
        endereco = novoEndereco;  
    }  
    public string ObterNome(){  
        return nome;  
    }  
}
```



# Exercício

```
class Agenda{
    private Pessoa[] pessoas = new Pessoa[5];
    int quantidadePessoas = 0;
    public void InserirPessoa(string nome, string endereco)
    {
        Pessoa pessoa = new Pessoa();
        pessoa.InformarDados(nome, endereco);
        pessoas[quantidadePessoas] = pessoa;
        quantidadePessoas++;
    }

    public Pessoa ObterPessoa(int posicao)
    {
        if (posicao < quantidadePessoas){
            Pessoa p = pessoas[posicao];
            return p;
        }
        else{
            Console.WriteLine("Erro!!! A posição informada não contém uma pessoa");
            return null;
        }
    }
}
```



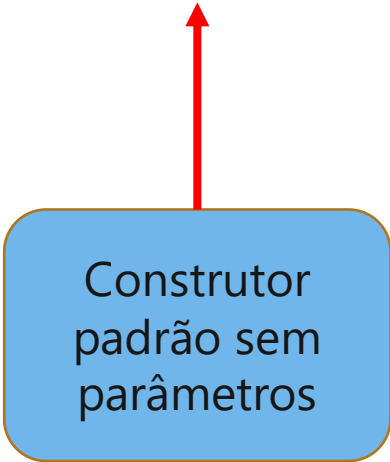
# Exercício

```
class Program{  
    public static void Main(){  
        Agenda agenda = new Agenda();  
        agenda.InserirPessoa("Fulano de Tal", "Rua X");  
        Pessoa pessoa = agenda.ObterPessoa(0);  
        Console.WriteLine(pessoa.ObterNome() + " - " + pessoa.endereco);  
    }  
}
```



# Construtor

- É um método chamado sempre que o Objeto é instanciado (declarado);
  - Pessoa fulano\_de\_tal = new Pessoa();

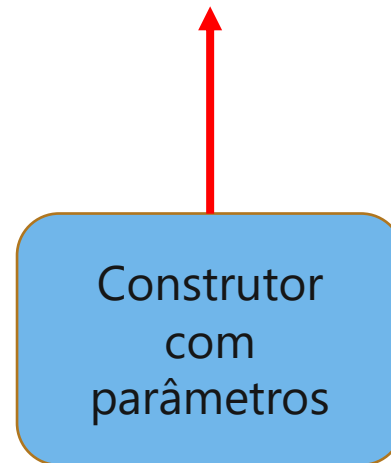


Construtor  
padrão sem  
parâmetros



# Construtor

- É um método chamado sempre que o Objeto é instanciado (declarado);
  - Pessoa fulano\_de\_tal = new Pessoa(“Fulano\_de\_tal”, “Rua X”);



# Exercício

- Refaça o método inserir pessoa da classe agenda com a nova definição da classe pessoa:

```
class Pessoa{  
    //declaração dos atributos  
    private string nome;  
    public string endereco;  
  
    //Construtor  
    public Pessoa(string novoNome, string novoEndereco)  
    {  
        nome = novoNome;  
        endereco = novoEndereco;  
    }  
  
    public string ObterNome()  
    {  
        return nome;  
    }  
}
```



# Exercício

```
class Agenda{
    private Pessoa[] pessoas = new Pessoa[5];
    int quantidadePessoas = 0;
    public void InserirPessoa()
    {
        Pessoa pessoa = new Pessoa("Fulano de Tal", "Rua X");
        //pessoa.InformarDados(nome, endereco);
        pessoas[quantidadePessoas] = pessoa;
        quantidadePessoas++;
    }

    public Pessoa ObterPessoa(int posicao)
    {
        if (posicao < quantidadePessoas){
            Pessoa p = pessoas[posicao];
            return p;
        }
        else{
            Console.WriteLine("Erro!!! A posição informada não contém uma pessoa");
            return null;
        }
    }
}
```



# Exercício

```
class Program{  
    public static void Main(){  
        Agenda agenda = new Agenda();  
        agenda.InserirPessoa();  
        Pessoa pessoa = agenda.ObterPessoa(0);  
        Console.WriteLine(pessoa.ObterNome() + " - " + pessoa.endereco);  
    }  
}
```





# Métodos Estáticos

- É um método que não precisa de uma instância da classe para ser chamado
  - Usualmente, métodos definidos em uma classe são aplicados a objetos daquela classe
  - Existe no entanto situações nas quais um método pode fazer uso apenas dos recursos de uma classe (e não das informações associadas a cada objeto individualmente) para realizar sua tarefa
- Exemplos:

`Math.Sqrt(42);`

`Console.Write("Olá Mundo");`



# Métodos Estáticos

```
class Calculadora{  
    public static int VerificarPositivo(int numero){  
        if (numero < 0){  
            return 0;  
        }  
        else{  
            return 1;  
        }  
    }  
}
```

- Para chamar o método NÃO precisamos declarar um objeto da classe:

```
Console.WriteLine(Calculadora.VerificarPositivo(-1));
```



# Palavra-chave this

- **this** é usada para referenciar o objeto atual dentro de uma classe. Ela pode ser usada em diferentes contextos com as seguintes finalidades:

- 1) Diferir os parâmetros do método e os atributos da classe: O this pode ser usado para distinguir entre um parâmetro do método e um atributo com o mesmo nome. É útil quando há uma ambiguidade de nomes. Por exemplo:

```
class Exemplo{  
    private int valor;  
  
    public void SetValor(int valor){  
        this.valor = valor; // "this.valor" refere-se ao campo da  
                             // classe, enquanto "valor" refere-se ao parâmetro do método  
    }  
}
```



# Palavra-chave this

- 2) Acessar atributos ocultos: Quando uma classe herda outra classe e possui atributos com o mesmo nome, o this pode ser usado para acessar o atributo da classe atual, mesmo que esteja oculto pela classe base. Por exemplo:

```
class Pessoa{
    protected string nome;

    public Pessoa(string nome){
        this.nome = nome; // "this.nome" refere-se ao atributo da classe
    }
}
```

```
class Cliente : Pessoa{
    private string nome; // Oculta o campo "nome" da classe base

    public Cliente(string nome) : base(nome)
    {
        this.nome = nome; // "this.nome" refere-se ao campo da classe
    }
}
```

Herança ainda vão  
estudar  
futuramente



# Palavra-chave this

- 3) Referenciar atributos da classe atual: O this pode ser usado para referenciar outros atributos da classe atual, como métodos e atributos, dentro de outros métodos ou construtores da mesma classe. Por exemplo:

```
class Exemplo{
    private int valor;

    public Exemplo(int valor){
        this.valor = valor;
    }

    public void ImprimirValor(){
        Console.WriteLine(this.valor); // "this.valor" refere-se ao
campo "valor" da classe
    }
}
```



# Palavra-chave params

- ***params*** é usado para definir um parâmetro do método que permite que um número variável de argumentos seja passado para o método. Assim, é possível fornecer zero ou mais argumentos para o parâmetro marcado com params.
- O params é usado na declaração do último parâmetro do método, seguido por um array unidimensional do tipo desejado e precedido por um modificador de acesso. A sintaxe é a seguinte:

```
public void Metodo(params Tipo[] parametros){  
    // Corpo do método  
}
```



# Palavra-chave params

- ***params: como usar?***

```
public void ImprimirValores(params int[] valores){  
    foreach (int valor in valores){  
        Console.WriteLine(valor);  
    }  
}
```

```
// Chamando o método com diferentes quantidades de argumentos  
ImprimirValores(); // Sem argumentos  
ImprimirValores(1); // Um argumento  
ImprimirValores(1, 2, 3); // Três argumentos
```

**OBS:** O params só pode ser usado para o último parâmetro do método. Além disso, um método só pode ter um parâmetro params.



# Exercício Classe

- Crie uma classe Aluno em C# com os seguintes (atributos) públicos:
  - nome, curso e período;
  - Crie um método para imprimir os dados do Aluno;
- Crie um programa para criar 3 objetos do tipo Aluno e solicite ao usuário que entre com dados para os três Alunos criados;
- Ao final imprima os valores fornecidos para esses alunos;





# Exercício Classe

- Altere a classe Aluno da seguinte forma:
  - Altere os atributos para privados (private) e crie métodos para atribuir e obter os valores dos atributos;
  - Crie outros 2 atributos (privados) para armazenar a nota do aluno em 2 disciplinas;
  - Crie um método que retorne a média das notas do aluno;
  - Crie um método que altera o período (soma mais um) do caso a média seja maior ou igual a 60
  - Altere o programa que solicita os dados dos alunos, agora solicite os dados de 40 alunos.



# Exercício Classe

- Crie uma classe Professor em C# com os seguintes atributos:
  - Nome e Salario;
  - Crie um construtor que recebe o Nome do professor;
  - Crie um método para aumentar o salário do professor dado um percentual informado como parâmetro;
  - Crie um método para imprimir o Nome e o Salário do Professor;
  - Crie um programa para testar a classe Professor;



# Referências

- Adaptado do material do profº. Ismael Santana Silva;
- ASCENCIO et al. Fundamentos da programação de computadores: algoritmos, Pascal e C/C++ e Java. 3.ed. São Paulo: Pearson Prentice Hall, 2008.

